

Beschreibung

Verfahren, Anordnung sowie ein Satz mehrerer Anordnungen zum Schutz mehrerer Programme und/oder mehrerer Dateien vor einem unbefugten Zugriff durch einen Prozeß

Die Erfindung betrifft ein Verfahren, eine Anordnung sowie ein Satz mehrerer Anordnungen zum Schutz mehrerer Programme vor einem unbefugten Zugriff durch einen Prozeß.

Ein Verfahren und eine Anordnung zum Schutz mehrerer Programme vor einem unbefugten Zugriff durch einen Anwender ist aus [1] oder [6] bekannt. Der Zugriffsschutz für ein Programm ist bei dem Verfahren aus [1] dadurch realisiert, daß jedem Benutzer eines Systems eine Zugriffsberechtigungsdatei zugeordnet wird. Versucht ein Prozeß, auf ein Programm zuzugreifen, so wird überprüft, ob der Benutzer, der den Prozeß gestartet hat, das Recht hat, auf das entsprechende Programm zuzugreifen. Der Zugriff wird nur gestattet, wenn der Prozeß von einem befugten und somit mit den Zugriffsrechten ausgestatteten Benutzer gestartet wurde.

Aus [2] ist ein sogenannter VirensScanner bekannt. Ein VirensScanner überprüft die gespeicherte, bekannte Folge von Daten, durch die das Programm realisiert ist. Wird eine Abweichung gegenüber der bekannten Folge festgestellt, so wird ein Benutzer des Systems benachrichtigt, daß möglicherweise das System mit einem Virus behaftet ist.

Aus [1] ist ferner ein Betriebssystem für einen Rechner bekannt. Das aus [1] bekannte Betriebssystem weist verschiedene Sicherheitslücken auf, durch die es einem Angreifer möglich ist, die Integrität von Programmen, die unter Verwendung des Betriebssystems durchgeführt werden, zu gefährden.

Ein möglicher Mechanismus, um den Schutz der Programme bei Verwendung dieses Betriebssystems zu gefährden ist ebenfalls in [5] beschrieben.

- 5 In [7] ist ein Computersystem zur Lizensierung von Software beschrieben.

Somit liegt der Erfindung das Problem zugrunde, mehrere Programme und/oder mehrere Dateien vor einem unbefugten Zugriff durch einen Prozeß zu schützen unter Verwendung eines Betriebssystems, welches grundsätzlich Sicherheitslücken aufweist.

Das Problem wird durch das Verfahren sowie durch die Anordnung gemäß den Merkmalen der unabhängigen Patentansprüche gelöst.

Bei einem Verfahren zum Schutz mehrerer Programme und/oder mehrerer Dateien vor einem unbefugten Zugriff durch einen Prozeß, ist jedem zu schützenden Programm und/oder jeder zu schützenden Datei jeweils ein Adressraum zugeordnet. Jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist ferner jeweils eine Prozeß-Datei zugeordnet, wobei in einer Prozeß-Datei gespeichert ist, welcher Prozeß oder welche Prozesse in dem jeweiligen Adressraum ablaufen darf oder dürfen. Während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei für einen Prozeß, der auf den Adressraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will wird überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist. Für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, wird der zugreifende Prozeß gestartet; sonst wird der zugreifende Prozeß nicht gestartet.

- 35 Bei einem weiteren Verfahren zum Schutz mehrerer Programme und/oder mehrerer Dateien vor einem unbefugten Zugriff durch einen Prozeß, ist jedem zu schützenden Programm und/oder je-

- der zu schützenden Datei jeweils ein Adreßraum zugeordnet.
Jedem zu schützenden Programm und/oder jeder zu schützenden
Datei ist ferner jeweils eine Prozeß-Datei zugeordnet, wobei
in einer Prozeß-Datei gespeichert ist, welcher Prozeß oder
welche Prozesse in dem jeweiligen Adreßraum ablaufen darf
oder dürfen. Während des Ablaufs eines zu schützenden Pro-
gramms und/oder einer zu schützenden Datei für einen Prozeß,
der auf den Adreßraum des zu schützenden Programms und/oder
einer zu schützenden Datei zugreifen will wird überprüft, ob
der zugreifende Prozeß in der entsprechenden Prozeß-Datei an-
gegeben ist. Für den Fall, daß der zugreifende Prozeß in der
Prozeß-Datei angegeben ist, wird der zugreifende Prozeß wei-
tergeführt; sonst wird der zugreifende Prozeß beendet.
- Eine Anordnung zum Schutz mehrerer Programme vor einem unbefugten Zugriff durch einen Prozeß, weist einen Prozessor auf,
der derart eingerichtet ist, daß folgende Schritte durchführbar sind:
- jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils ein Adreßraum zugeordnet,
 - jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils eine Prozeß-Datei zugeordnet,
 - in einer Prozeß-Datei ist gespeichert, welcher Prozeß oder welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,
 - während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei wird für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will, überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,
 - für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, wird der zugreifende Prozeß gestartet, und
 - sonst wird der zugreifende Prozeß nicht gestartet.

35

Eine weitere Anordnung zum Schutz mehrerer Programme vor ei-
nem unbefugten Zugriff durch einen Prozeß, weist einen Pro-

zessor auf, der derart eingerichtet ist, daß folgende Schritte durchführbar sind:

- jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils ein Adreßraum zugeordnet,

5 - jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils eine Prozeß-Datei zugeordnet,

- in einer Prozeß-Datei ist gespeichert, welcher Prozeß oder welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,

10 - während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei wird für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,

15 - für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, wird der zugreifende Prozeß weitergeführt, und

- sonst wird der zugreifende Prozeß beendet.

20 Ein Satz mehrerer Anordnungen und eine mit jeder Anordnung des Satzes mehrerer Anordnungen verbundene Server-Anordnung zum Schutz mehrerer Programme vor einem unbefugten Zugriff durch einen Prozeß ist derart eingerichtet, daß jede Anordnung einen Prozessor aufweist, der derart einge-

25 richtet ist, daß folgende Schritte durchführbar sind:

- jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils ein Adreßraum zugeordnet,

- jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils eine Prozeß-Datei zugeordnet,

30 - in einer Prozeß-Datei ist gespeichert, welcher Prozeß oder welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,

- während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei wird für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,

35

- für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, wird der zugreifende Prozeß gestartet oder weitergeführt, und

- sonst wird ein Alarmsignal generiert und an die Server-Anordnung gesendet,

5 und die Server-Anordnung einen Prozessor aufweist, der derart eingerichtet ist, daß abhängig von mindestens einem empfangenen Alarmsignal eine vorgegebene Aktion ausgelöst wird.

10 Unter Adreßraum ist in diesem Zusammenhang ein Programm-Bereich zu verstehen, der jeweils einem Programm zugeordnet ist.

Durch die Erfindung werden mehrere Sicherheitslücken des in
15 [1] beschriebenen Betriebssystems geschlossen.

Weiterhin wird das jeweils zu schützende Programm gegen einen prozeduralen Angriff (Angriff auf einen Prozeß), z.B. gegen ein Trojanisches Pferd, geschützt.

20 Ferner ist ein erheblicher Vorteil der Erfindung darin zu sehen, daß bei skalierbarem Aufwand ein definiertes Maß an Sicherheit für das zu schützende Programm gewährleistet werden kann.

25 Durch den Satz mehrerer Anordnungen, die jeweils mit der Server-Anordnung verbunden sind, ist ein Schutz lokal bei den Anordnungen möglich derart, daß bei einem erkannten Angriff ein Alarmsignal generiert und an die Server-Anordnung gesendet wird, in der zentral eine vorgegebene Aktion ausgeführt wird. Auf diese Weise ist die Entdeckung lokaler Prozesse möglich, die der Server-Anordnung selbst nicht bekannt sind.

Bevorzugte Weiterbildungen der Erfindung ergeben sich aus den
35 abhängigen Ansprüchen.

Es ist in einer Weiterbildung zur Erhöhung des erreichbaren Sicherheitsniveaus vorteilhaft, zumindest einem Teil der in einer Prozeß-Datei angegebenen Prozesse einen eindeutig kennzeichnenden kryptographischen Wert zu bilden, wobei der jeweilige Wert in der Prozeß-Datei enthalten ist. Für den zugreifenden Prozeß wird dessen kryptographischer Wert gebildet und bei der Überprüfung werden die kryptographischen Werte der Prozesse miteinander verglichen.

Der kryptographische Wert kann eine digitale Signatur sein. Er kann aber auch unter Verwendung einer Hash-Funktion, allgemein einer Einwegfunktion, gebildet werden.

In einer weiteren Ausgestaltung ist es vorteilhaft, in einem Aufrufmechanismus für eine Funktion eines Betriebssystemkerns, mit dem die Programme ausgeführt werden, einen Aufruf des zugreifenden Prozesses zu einer Überprüfungsfunktion weiterzuleiten, in der die Überprüfung erfolgt. Auf diese Weise ist eine effiziente und somit kostengünstige Realisierung der Erfahrung möglich.

Die Überprüfungsfunktion kann als dynamisch bindbare Datei in den überwachten Adressraum eingebunden werden, wodurch eine weitere Verbesserung in der Schutzwirkung erreicht wird.

Ein Aufruf eines zugreifenden Prozesses kann auch zu einer Überprüfungsfunktion, die in dem Betriebssystemkern integriert ist, weitergeleitet werden, wobei die Überprüfung in der Überprüfungsfunktion erfolgt. Auf diese Weise kann die erreichbare Sicherheit des Schutzes für die Programme noch weiter erhöht werden.

Eine weitere Erhöhung des erreichbaren Sicherheitsniveaus kann gewährleistet werden, wenn ein Schutzprogramm, welches derart eingerichtet ist, daß die Erfahrung ausführbar ist, verschlüsselt gespeichert ist und zu Beginn des Verfahrens entschlüsselt wird. Nach der Entschlüsselung des Schutzpro-

gramms kann dessen Integrität überprüft werden und das Verfahren wird nur dann ausgeführt, wenn die Integrität des Schutzprogrammes gewährleistet ist. Nach der Integritätsprüfung des Schutzprogramms kann die Integrität aller in den Prozeß-Dateien enthaltenen Prozesse überprüft werden und das Verfahren wird nur ausgeführt, wenn die Integrität des Schutzprogramms gewährleistet ist. Nach der Integritätsprüfung der Prozesse, kann die Integrität des zu schützenden Programms überprüft werden und das Programm sollte nur ausgeführt werden, wenn die Integrität des Schutzprogramms gewährleistet ist.

Die Erfindung ist vorteilhaft einsetzbar in dem in [1] beschriebenen Betriebssystem.

Obwohl das im weiteren erläuterte Ausführungsbeispiel den Schutz von Programmen beschreibt, so ist ebenfalls ein Schutz mehrerer Dateien ohne weiteres gemäß der gleichen Vorgehensweise möglich.

Ein Ausführungsbeispiel der Erfindung ist in den Figuren dargestellt und wird im weiteren näher erläutert.

Es zeigen
Figur 1 eine Skizze, in der das der Erfindung zugrundeliegende Prinzip symbolisch dargestellt ist,

Figur 2 eine Skizze, in der ein Prozeßschichtenmodell darstellt ist;

Figur 3 ein Blockdiagramm, in dem ein Rechnernetz dargestellt ist;

Figur 4 ein Ablaufdiagramm, in dem die einzelnen Verfahrensschritte des Ausführungsbeispiels dargestellt sind;

Figur 5 eine Skizze, in der das Prinzip einer möglichen Integration der Erfindung in einem Betriebssystem dargestellt ist;

5 Figur 6 eine Skizze, in der die mögliche Realisierung gemäß Figur 5 detailliert dargestellt ist;

Figur 7 eine Skizze, in der eine weitere mögliche Integration der Erfindung in ein Betriebssystem dargestellt ist.

10 Fig.3 zeigt einen ersten Rechner 301 mit einer Eingangs-/ Ausgangsschnittstelle 302, die über einen Bus 303 mit einem Speicher 304 und einem Prozessor 305 verbunden ist. Über die Eingangs-/Ausgangsschnittstelle 302 ist der erste Rechner 301 15 über ein Rechnernetz 306 mit einer Vielzahl von Rechnern 307, 308, 309, 310, 311 verbunden.

20 Das zur Übertragung digitaler Daten verwendete Kommunikationsprotokoll ist das TCP/IP-Protokoll (Transmission Control Protocol/Internet Protocol).

25 Die Erfindung schützt den ersten Rechner 301 vor unbefugten Zugriffen von Prozessen, die entweder in dem Speicher 304 des ersten Rechners 301 gespeichert sind oder von den weiteren Rechnern 307, 308, 309, 310, 311, die auf den ersten Rechner 301 zugreifen/einwirken.

30 In dem ersten Rechner 301 ist das in [1] beschriebene Betriebssystem implementiert.

Das im weiteren detailliert erläuterte Prinzip, welches dem Verfahren bzw. der Anordnung zugrunde liegt, ist anschaulich in Fig.1 dargestellt.

35 Symbolisch dargestellte Programme 101, 102, 103 sollen durch die Erfindung gegen unbefugten Zugriff durch mindestens einen

Prozeß 104, 105, 106, der oder die auf ein Programm 101, 102, 103 zugreifen wollen, geschützt werden.

Die Programme 101, 102, 103 und Prozesse 104, 105, 106 verwenden als Betriebssystem, dargestellt als eine die Programme 101, 102, 103 und Prozesse 104, 105, 106 umrahmende Einheit 107 das in [1] beschriebene Betriebssystem.

Durch das Verfahren bzw. durch die Anordnung wird anschaulich um jedes zu schützende Programm 101, 102, 103 eine programm-spezifische „Schutzhülle“ 108, 109, 110 gebildet. Durch die programmspezifische Sicherung wird ein frei skalierbares Sicherheitsniveau für das zu schützende Programm 101, 102, 103 erreicht.

Das Verfahren bzw. die Anordnung kann, wie in einem Prozeßschichtenmodell 201 in Fig.2 dargestellt, auf verschiedenen logischen Ebenen zu schützender Programme 101, 102, 103 realisiert werden. Fig.2 zeigt drei logische Ebenen in dem Prozeßschichtenmodell 201.

Die Sicherung vor einem Angreifer 205 kann auf der Ebene zu schützender Anwendungsprogramme 202, auf der Ebene zu schützender Betriebssystemprogramme 203 sowie auf der Ebene des Betriebssystemkerns, und auf der System-Hardware 207 erfolgen.

Je näher an der System-Hardware 207 die Sicherung eines Programms erfolgt, desto größer ist das Sicherheitsniveau, das durch die Erfindung erreicht wird.

Die Schutzhülle 206 wird zur Laufzeit des zu schützenden Programms 101, 102, 103 um das Programm „gelegt“.

Das Verfahren wird in Form zyklischer, nebenläufiger Prozesse realisiert. Das Verfahren wird anhand des Ablaufdiagramms, das in Fig.4 dargestellt ist, erläutert.

Als erstes nach dem Start des Betriebssystems (Schritt 401) wird ein Schutzprogramm gestartet (Schritt 402). Das Schutzprogramm ist derart eingerichtet, daß das im weiteren beschriebene Verfahren ausführbar ist. Das Schutzprogramm ist in verschlüsselter Form gespeichert, wodurch eine Veränderung des Schutzprogramms selbst nicht möglich ist.

Auch wird durch die Verschlüsselung des Schutzprogramms eine detaillierte Analyse des das Verfahren ausführenden Programms verhindert.

Zum Start des Schutzprogramms (Schritt 402) wird das Schutzprogramm durch eine vorgegebene Startroutine, die den zur Entschlüsselung des Schutzprogramms erforderlichen Schlüssel und eventuell weitere Grundfunktionen des Betriebssystems enthält, entschlüsselt, wodurch der eigentliche Programmcode des Schutzprogramms ermittelt wird.

Auf diese Weise ist das Schutzprogramm im aktiven Zustand vor off-line Angriffen wie disassemblieren, debuggen, patchen, ..., geschützt.

Nach der Entschlüsselung (Schritt 402) des Schutzprogramms wird die Integrität des Schutzprogramms dynamisch überprüft (Schritt 403).

Ist die Integrität des Schutzprogramms nicht gewährleistet, so wird das Verfahren abgebrochen (Schritt 404).

In einem weiteren Schritt wird die Integrität der Prozesse des Betriebssystems dynamisch überprüft (Schritt 405). Bei negativer Integritätsprüfung wird das Verfahren wiederum abgebrochen (Schritt 404).

Ist die Integrität der Prozesse des Betriebssystems 405 gewährleistet, so wird das zu schützende Programm gestartet.

Das oben beschriebene Verfahren wird für jedes zu schützende Programm 101, 102, 103, dynamisch durchgeführt.

- 5 Jedem zu schützenden Programm 101, 102, 103 ist jeweils eine Prozeßdatei 111, 112, 113 zugeordnet.

In einer Prozeßdatei 111, 112, 113 ist für das zu schützende Programm 101, 102, 103, dem die Prozeß-Datei 111, 112, 113 zugeordnet ist, angegeben, welche Prozesse in einem Adreßraum, der ebenfalls jedem Programm 101, 102, 103 eindeutig zugeordnet ist, ablaufen dürfen. In den Prozeß-Dateien sind die Angaben mittels einer den jeweiligen Prozeß eindeutig kennzeichnenden Hash-Funktion gespeichert.

- 15 Nach Start des jeweiligen Programms 101, 102, 103 (Schritt 407) wird die Integrität des Programms 101, 102, 103 selbst überprüft (Schritt 408).

- 20 Bei negativer Integritätsprüfung wird wiederum das Verfahren abgebrochen (Schritt 404).

Bei positiver Integritätsprüfung, d.h. wenn die Integrität des Programms 101, 102, 103 gewährleistet ist, wird das Verfahren für das zu schützende Programm 101, 102, 103 solange wiederholt, bis das zu schützende Programm selbst beendet wird (Schritt 409).

- 30 Das Verfahren wird abhängig von einem vorgebbaren Ereignis oder in einem vorgebbaren Zeitabstand zwischen zwei Ausführungen des Verfahrens iteriert (Schritt 410).

Sobald ein Prozeß 104, 105, 106 auf den Adreßraum bzw. das Programm 101, 102, 103 selbst zugreifen will, wird in einem weiteren Schritt (Schritt 411) überprüft, ob der Prozeß 104, 105, 106 in der Prozeß-Datei des zu schützenden Programms

101, 102, 103, auf das der Prozeß 104, 105, 106 zugreifen will, enthalten ist oder nicht.

Dies erfolgt durch Bildung eines Hash-Wertes über den zugreifenden Prozeß 104, 105, 106 und Vergleich des Hash-Wertes des zugreifenden Prozesses 104, 105, 106 mit den Hash-Werten, die in der Prozeß-Datei gespeichert sind. Ist der zugreifende Prozeß in der Prozeß-Datei des zu schützenden Programms 101, 102, 103 angegeben, so wird der Prozeß 104, 105, 106 durchgeführt (Schritt 412).

Sonst wird der Prozeß 104, 105, 106 nicht gestartet (Schritt 413) und der Benutzer wird über einen möglichen Angriff auf sein Programm 101, 102, 103 informiert.

In vorgebbaren Zeitabständen oder ereignisgesteuert wird für jeden in dem Adressraum eines Programms 101, 102, 103 aktiven Prozeß, d.h. ablaufenden Prozeß, überprüft, ob der jeweilige Prozeß in der Prozeß-Datei des entsprechenden Programms 101, 102, 103, dessen Adressraum untersucht wird, enthalten ist. Ist dies nicht der Fall, so wird der entsprechende Prozeß beendet und der Benutzer wird auf einen möglichen Angriff auf sein Programm 101, 102, 103 hingewiesen.

Auf diese Weise ist eine regelmäßige Überwachung des Programms gewährleistet.

Im weiteren werden Möglichkeiten zur Integration des oben beschriebenen Verfahrens in das in [3] beschriebene Betriebssystem dargestellt:

1. Möglichkeit:

Integration einer dynamisch bindbaren Datei 501 in die Anwendungs-Programmier-Schnittstelle (Application Programming Interface, API) (vgl. Fig.5).

Die dynamisch bindbare Datei 501 wird im Adreßraum des potentiellen Angreifer-Programms 502 aktiv. Unter Verwendung der dynamisch bindbaren Datei 501 werden folgende Schritte in dem Adreßraum des potentiellen Angreifer-Programms 502 ausge-

5 führt:

- Von der dynamisch bindbaren Datei 501 werden alle Kennzeichnungen (Modulhändel) von jeder weiteren dynamisch bindbaren Datei ermittelt, die zu überwachende Schnittstellen-Aufrufe enthält. Damit können alle Zugriffe auf alle zu 10 überwachenden dynamisch bindbaren Dateien ermöglicht werden.

Als für Angriffe relevant werden alle Schnittstellen-Aufrufe zum direkten oder indirekten Starten, Beenden und Kontrollieren von Prozessen, z.B. schreibende Zugriffe auf den Prozeßspeicher, Änderungen der Zugriffsrechte, prinzipiell alle Schnittstellen-Befehle, die mit fremden Prozeßmarkierungen (Prozeßhandles) arbeiten, alle Befehle zum Realisieren von Message Hooks (eine programmierbare Filterfunktion für Nachrichten zur GUI-Interkommunikation (Graphic User Interface) und zur Prozeßinterkommunikation), und für Debugging-Zwecke betrachtet.

Unter dem Ausdruck „indirektes Starten“ werden auch Mechanismen eines Compilers zur Selbstmodifikation von Programmcode, OLE (Object Link Embedding) und RPC-Mechanismen (Remote Procedure Call) sowie Zugriffe aus anderen Betriebssystem-Programmier-Schnittstellen verstanden. Ferner umfaßt dieser Ausdruck auch die Kontrolle der Mechanismen, die für den Ablauf dynamisch bindbarer Dateien 501 (ActiveX-Befehle usw.) eingesetzt werden.

- 30
- Ferner werden der dynamischen bindbaren Datei 501 Schreibrechte für den Adreßraum zugeordnet.
 - Anstelle des Original-Schnittstellen-Aufrufs wird ein Sprung-Befehl gespeichert und die ersetzen Befehle des Original-Codes 503 werden gesichert.

35

Versucht nun ein Prozeß 504 über einen Schnittstellenaufruf APIFktCall() auf das zu schützende Programm 502 zuzugreifen

bzw. ein bisher inaktives Programm zu starten, so wird durch die dynamisch bindbare Datei 501 das Schutzprogramm 506 aufgerufen. Als Übergabeparameter für das Schutzprogramm 506 wird dem Schutzprogramm 506 angegeben, welches Programm 502 5 durch den zugreifenden Prozeß 504 aufgerufen werden soll.

Durch einen Vergleich mit in der Prozeß-Datei des jeweiligen Programms 502 angegebenen zugelassenen Schnittstellen-Aufrufen wird dann gemäß dem oben beschriebenen Verfahren 10 entschieden, ob der Schnittstellen-Aufruf 505 zugelassen wird oder nicht.

Der zugreifende Prozeß 504 wird dann entweder ausgeführt oder „abgeblockt“. Wird in dem Schutzprogramm 506 entschieden, daß 15 der zugreifende Prozeß 504 ausgeführt werden soll, so wird der Original-Code des Schnittstellen-Aufrufs 503 ausgeführt und nach dessen Ausführung wird ein Return-Code 507 an den zugreifenden Prozeß 504 zurückgegeben. Sonst wird eine Fehlermeldung 508 an den zugreifenden Prozeß 504 gemeldet.

20 Das oben beschriebene Verfahren verwendet den grundsätzlichen Mechanismus der sogenannten DLL-Injektion, die aus [5] bekannt ist.

25 2. Möglichkeit:

Fig.6 zeigt das in Fig.5 beschriebene Prinzip in verfeinerter Realisierung. Diese Variante eignet sich insbesondere für den Fall, daß sich fälschlicherweise ein als „sicher“ deklarierter Prozeß 30 in dem Adressraum befindet, der die dynamisch bindbare Datei 601 selbst angreift.

Durch das im folgenden beschriebene Verfahren wird gewährleistet, daß ein Zugriff auf vorgegebene Daten nur aus der dynamisch bindbaren Datei 601 selbst heraus möglich ist und Zugriffe aus einem anderen Befehlsfolgesegment, insbesondere einer angreifenden Applikation, verhindert wird.

Das im folgenden dargestellte Verfahren wird vorzugsweise mit statischer Modifikation des oben beschriebenen Verfahrens realisiert.

5

Für das im weiteren beschriebene Verfahren werden folgende Annahmen getroffen:

- Zu schützende Daten werden in einem geschützten Bereich 602 gespeichert, die bei der Initialisierung der dynamisch bindbaren Datei 601 angelegt und beschrieben wird und anschließend ein Schutzattribut erhält, so daß auf den geschützten Bereich 602 nur durch die dynamisch bindbare Datei 601 selbst zugegriffen werden kann.

10

- Alle Bereiche, die ausführbaren Code enthalten, erhalten das Schutzattribut „page_execute“, wodurch verhindert wird, daß der entsprechende ausführbare Code nicht verändert werden kann, ohne daß das Schutzattribut zuvor geändert wird.
- Jede Schnittstellen-Funktion 603 wird auf folgende Weise gesichert: Eine Einsprungsadresse für die Schnittstellen-Funktion 603 wird durch eine modifizierte Einsprungsadresse, die zu einer modifizierten Schnittstellen-Funktion 604 führt, ersetzt.

15

20

In der modifizierten Schnittstellen-Funktion wird verzweigt zu einem Schnittstellen-Prozeß 605, der in der dynamisch bindbaren Datei 601 enthalten ist. Dieser Schnittstellen-Prozeß verzweigt zu dem Schutzprogramm 606, durch das für einen aufrufenden Prozeß 607, der mit einem Schnittstellenaufruf 608 versucht, auf die Schnittstellenfunktion 603 zugreifen, überprüft, ob dieser Aufruf für den zugreifenden Prozeß 607 zugelassen ist.

25

30

Ist dies der Fall, so wird die Schnittstellenfunktion 603 ausgeführt und es wird nach Durchführung der Schnittstellenfunktion 603 wiederum in die modifizierte Schnittstellenfunktion 604 verzweigt, was durch einen Pfeil 609 symbolisiert ist. Nach Ausführung weiterer vorgebbarer Befehle wird in eine Schnittstellen-Rückkehr-Funktion 610 verzweigt, was durch einen Pfeil 611 angedeutet ist. Dies er-

35

folgt durch einen Sprungbefehl. In der Schnittstellen-Rückkehr-Funktion 610 wird noch einmal überprüft (Schritt 612), ob der Schnittstellen-Aufruf 608 zugelassen ist. Wenn dies nicht der Fall ist, wird eine Fehlermeldung 613 an den Prozeß 607 gesendet. Dies erfolgt ebenso, wenn in 5 der Schnittstellen-Funktion 605 unter Verwendung des oben beschriebenen Verfahrens ermittelt wurde, daß der Schnittstellen-Aufruf nicht zugelassen ist.

Wird jedoch auch in dem Überprüfungsschritt 612 in der 10 Schnittstellen-Rückkehr-Funktion 610 ermittelt, daß der Schnittstellen-Aufruf zugelassen ist, so wird das Ergebnis der aufgerufenen Schnittstellen-Funktion an den zugreifenden Prozeß 607 gesendet (Schritt 614).

15 3. Möglichkeit:

Die Erfindung kann auch in dem Betriebssystemkern integriert werden. Durch diese Ausführungsform wird erreicht, daß ein Kontrollmechanismus integrierbar ist, dessen Umgehung unter 20 Benutzerzugriffsrechten nicht mehr möglich ist, sondern nur unter den Zugriffsrechten des Systemadministrators. Damit wird das erreichbare Sicherheitsniveau erheblich gesteigert.

Ein hierzu verwendbarer Integrationsmechanismus ist in [4] 25 beschrieben. Bei diesem Mechanismus werden Schnittstellen-Aufrufe beim Übertritt in den Modus des Betriebssystemkerns (Kernel-Modus) im Betriebssystemkern selbst "abgefangen". Der Verwaltungsmechanismus für die Unterbrechnungs-Routine ist in diesem Fall im Betriebssystemkern realisiert und ist daher 30 gegen Zugriffe von Prozessen, die im Benutzermodus aktiv sind, geschützt. Eine Übersicht verschiedener solcher möglichen alternativen Implementierungsmöglichkeiten ist in [4] zu finden.

35 Fig. 7 zeigt in einer Übersicht zwei oben beschriebene Möglichkeiten zur Realisierung.

Ein Anwendungsprogramm 701 (Applikation) verwendet zum Programmablauf Funktionen des Betriebssystems.

Die Funktionen des Betriebssystems sind gruppiert in Funktionen 702 des Betriebssystems in einem Benutzermodus (User-Mode) und in Funktionen 703 des Kernel-Modus. Die Funktionen sind symbolisch jeweils als Blöcke dargestellt.

Durch dynamisch bindbare Dateien (*.dll) ist es möglich, das Verfahren im Rahmen des Benutzermodus zu integrieren, was durch einen ersten Block 704 unter Verwendung einer dynamisch bindbaren Datei "NTDLL.DLL" 705 des in [1] beschriebenen Betriebssystems erfolgen kann.

Die weitere Integrationsmöglichkeit des Verfahrens in den Betriebssystemkern ist durch einen zweiten Block 706 angedeutet, wobei bei dieser Integrationsvariante der Mechanismus beim Übergang des Benutzermodus in den Kernel-Modus integriert wird.

Im weiteren werden einige Alternativen zu den oben erläuterten Ausführungsbeispielen dargestellt:

Der Schutz des Schutzprogramms kann dadurch erhöht werden, daß die Startroutine im Betriebssystemkern integriert ist, z.B. als sogenannter Kernel-Modus-Prozeß oder auch System Service.

Die dynamisch bindbare Datei kann auch sowohl statisch als auch dynamisch, d.h. lediglich während der Laufzeit des zu schützenden Programms 502 oder während der gesamten Laufzeit des Betriebssystems vorgesehen sein.

Auch können alternativ Software-Unterbrechungs-Routinen als Alternative zu der dynamisch bindbaren Datei verwendet werden.

Die Erfindung kann sowohl durch Software als auch durch Hardware oder zum Teil durch Software und zum Teil durch Hardware realisiert werden.

5 Ferner kann, wie in Fig.4 dargestellt, bei negativer Integritätsprüfung (Schritt 408) in einem weiteren Überprüfungs-
schrift (Schritt 414) überprüft werden, ob ein Nachladen des
originalen Schutzprogramms und/oder eines zu schützenden Pro-
gramms von einer vertrauenswürdigen Instanz möglich ist, oder
10 ob eine Wiederherstellung des Schutzprogramms und/oder des zu
schützenden Programms möglich ist, derart, daß dessen/deren
Integrität gewährleistet ist.

Ist dies nicht möglich, so wird das Verfahren abgebrochen
15 (Schritt 404).

Ist dies jedoch möglich, so wird die Integrität des neu gela-
denen bzw. wiederhergestellten Schutzprogramms weiter dyna-
misch überprüft (Schritt 403).

20 In einer weiteren Variante ist es vorgesehen, daß die Rechner
mit einem Server, in Fig.1 der erste Rechner, verbunden sind.

25 In jedem Rechner wird das oben beschriebene Verfahren zum
Schutz eines Programms und/oder einer Datei durchgeführt.

Für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei
nicht angegeben ist, wird ein Alarmsignal generiert und an
den Server gesendet. In dem Server wird abhängig von minde-
30 stens einem empfangenen Alarmsignal eine vorgegebene Aktion
ausgelöst wird, beispielsweise ein zentral gesteuerter Ab-
bruch eines Prozesses.

Im Rahmen dieses Dokuments wurden folgende Veröffentlichungen zitiert:

- [1] Microsoft Windows NT workstation resource kit:
5 comprehensive resource guide and utilities for
Windows NT 4.0, ISBN 1-57231-343-9, Microsoft Press,
C. Doyle (ed.), S. 117 - 118, 1996
- [2] Dr. Solomon's Anti-Virus Toolkit Workstation, öffentlich
10 zugänglich im Internet am 03.07.1998 unter der Internet-
Adresse:
<http://www.drsolomon.de/produkte/avtkws/avtkws.html>
- [3] M. Petrek, Under the hood, MS-Journal, No. 12, December
15 1996,
- [4] M. Russinovich, Windows NT System-Call Hooking,
Dr. Dobb's Journal, S. 42 - 46, Januar 1997,
- 20 [5] J. Richter, Advanced Windows, ISBN 1-573231-548-2,
3rd Edition, S. 899 ff., Kapitel: Breaking Through
Process-Boundary Walls, 1997
- [6] US 5 390 310
- 25 [7] US 5 023 907

Patentansprüche

1. Verfahren zum Schutz mehrerer Programme und/oder mehrerer Dateien vor einem unbefugten Zugriff durch einen Prozeß,
 - 5 - bei dem jedem zu schützenden Programm und/oder jeder zu schützenden Datei jeweils ein Adreßraum zugeordnet ist,
 - bei dem jedem zu schützenden Programm und/oder jeder zu schützenden Datei jeweils eine Prozeß-Datei zugeordnet ist,
 - bei dem in einer Prozeß-Datei gespeichert ist, welcher Prozeß oder welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,
 - 10 - bei dem während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will überprüft wird, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,
 - 15 - bei dem für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, der zugreifende Prozeß gestartet wird, und
 - 20 - sonst der zugreifende Prozeß nicht gestartet wird.
2. Verfahren zum Schutz mehrerer Programme und/oder mehrerer Dateien vor einem unbefugten Zugriff durch einen Prozeß,
 - 25 - bei dem jedem zu schützenden Programm und/oder jeder zu schützenden Datei jeweils ein Adreßraum zugeordnet ist,
 - bei dem jedem zu schützenden Programm und/oder jeder zu schützenden Datei jeweils eine Prozeß-Datei zugeordnet ist,
 - bei dem in einer Prozeß-Datei gespeichert ist, welcher Prozeß oder welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,
 - 30 - bei dem während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder einer zu schützenden Datei zugreifen will überprüft wird, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,

- bei dem für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, der zugreifende Prozeß weitergeführt wird, und
- sonst der zugreifende Prozeß beendet wird.

5

3. Verfahren nach Anspruch 1 oder 2,

- bei dem für zumindest einen Teil der in einer Prozeß-Datei angegebenen Prozesse ein den Prozeß eindeutig kennzeichnender kryptographischer Wert gebildet wird,

10 - bei dem in der Prozeß-Datei jeweils der kryptographische Wert eines Prozesses enthalten ist,

- bei dem für den zugreifenden Prozeß dessen kryptographischer Wert gebildet wird, und

15 - bei dem bei der Überprüfung die kryptographischen Werte der Prozesse miteinander verglichen werden.

4. Verfahren nach einem der Ansprüche 1 bis 3,

bei dem in einem Aufrufmechanismus für eine Funktion eines Betriebssystemkerns, mit dem die Programme ausgeführt werden,
20 ein Aufruf des zugreifenden Prozesses einer Überprüfungsfunktion zugeleitet wird, in der die Überprüfung erfolgt.

5. Verfahren nach Anspruch 4,

bei dem die Überprüfungsfunktion als dynamisch bindbare Datei
25 in den Adressraum eingebunden wird.

6. Verfahren nach einem der Ansprüche 1 bis 3,

30 - bei dem ein Aufruf des zugreifenden Prozesses einer Überprüfungsfunktion zugeleitet wird, in der die Überprüfung erfolgt, und

- bei dem die Überprüfungsfunktion in einen Betriebssystemkern eines Betriebssystems, mit dem die Programme ausgeführt werden, integriert ist.

35 7. Verfahren nach einem der Ansprüche 1 bis 6,

bei dem das Betriebssystem Windows NT ist.

8. Verfahren nach einem der Ansprüche 1 bis 7,
bei dem in einem vorgebbaren Zeitabstand für jeden aktiven Prozeß, der zusammen mit einem zu schützenden Programm und/oder einer zu schützenden Datei abläuft, überprüft wird, ob der aktive Prozeß in der Prozeß-Datei, die dem zu schützenden Programm und/oder der zu schützenden Datei zugeordnet ist, enthalten ist und der Prozeß beendet wird, falls dies nicht der Fall ist.
9. Verfahren nach einem der Ansprüche 1 bis 8,
bei dem abhängig von einem vorgebbaren Ereignis für jeden aktiven Prozeß, der zusammen mit einem zu schützenden Programm und/oder einer zu schützenden Datei abläuft, überprüft wird, ob der aktive Prozeß in der Prozeß-Datei, die dem zu schützenden Programm zugeordnet ist, enthalten ist und der Prozeß beendet wird, falls dies nicht der Fall ist.
10. Verfahren nach einem der Ansprüche 1 bis 9,
bei dem ein Schutzprogramm, welches derart eingerichtet ist, daß das Verfahren ausführbar ist, verschlüsselt gespeichert ist und zu Beginn des Verfahrens entschlüsselt wird
11. Verfahren nach einem der Ansprüche 1 bis 10,
bei dem die zu schützenden Programme und/oder die zu schützenden Dateien, verschlüsselt gespeichert sind und zu Beginn des Verfahrens entschlüsselt werden.
12. Verfahren nach Anspruch 10,
bei dem nach der Entschlüsselung des Schutzprogramms dessen Integrität überprüft wird und das Verfahren nur ausgeführt wird, wenn die Integrität des Schutzprogramms gewährleistet ist.
13. Verfahren nach Anspruch 12,
bei dem nach der Integritätsprüfung des Schutzprogramms die Integrität aller in den Prozeß-Dateien enthaltenen Prozesse

überprüft wird und das Verfahren nur ausgeführt wird, wenn die Integrität des Schutzprogramms gewährleistet ist.

14. Verfahren nach Anspruch 13,
5 bei dem nach der Integritätsprüfung der Prozesse die Integrität des zu schützenden Programms und/oder der zu schützenden Datei überprüft wird und das Verfahren nur ausgeführt wird, wenn die Integrität des Schutzprogramms gewährleistet ist.
- 10 15. Verfahren nach Anspruch 13 oder 14,
bei dem mindestens eine der Integritätsprüfung unter Verwendung eines kryptographischen Verfahrens erfolgt.
- 15 16. Verfahren nach einem der Ansprüche 3 bis 15,
bei dem der kryptographische Wert durch Anwendung einer Hash-Funktion gebildet wird
- 20 17. Anordnung zum Schutz mehrerer Programme vor einem unbefugten Zugriff durch einen Prozeß,
mit einem Prozessor, der derart eingerichtet ist, daß folgende Schritte durchführbar sind:
 - jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils ein Adreßraum zugeordnet,
 - jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils eine Prozeß-Datei zugeordnet,
 - in einer Prozeß-Datei ist gespeichert, welcher Prozeß oder welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,
 - während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei wird für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,
 - für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, wird der zugreifende Prozeß gestartet, und
 - sonst wird der zugreifende Prozeß nicht gestartet.

18. Anordnung zum Schutz mehrerer Programme vor einem unbefugten Zugriff durch einen Prozeß,
mit einem Prozessor, der derart eingerichtet ist, daß folgende Schritte durchführbar sind:
- jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils ein Adreßraum zugeordnet,
 - jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils eine Prozeß-Datei zugeordnet,
 - in einer Prozeß-Datei ist gespeichert, welcher Prozeß oder welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,
 - während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei wird für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,
 - für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, wird der zugreifende Prozeß weitergeführt, und
 - sonst wird der zugreifende Prozeß beendet.

19. Anordnung nach Anspruch 17 oder 18,
bei der der Prozessor derart eingerichtet ist, daß
- für zumindest einen Teil der in einer Prozeß-Datei angegebenen Prozesse ein den Prozeß eindeutig kennzeichnender kryptographischer Wert gebildet wird,
 - in der Prozeß-Datei jeweils der kryptographische Wert eines Prozesses enthalten ist,
 - für den zugreifenden Prozeß dessen kryptographischer Wert gebildet wird, und
 - bei der Überprüfung die kryptographischen Werte der Prozesse miteinander verglichen werden.
20. Anordnung nach einem der Ansprüche 17 bis 19,
bei der der Prozessor derart eingerichtet ist, daß in einem Aufrufmechanismus für eine Funktion eines Betriebssystem-

kerns, mit dem die Programme ausgeführt werden, ein Aufruf des zugreifenden Prozesses einer Überprüfungsfunction zugeleitet wird, in der die Überprüfung erfolgt.

- 5 21. Anordnung nach einem der Ansprüche 17 bis 20, bei der der Prozessor derart eingerichtet ist, daß das Betriebssystem Windows NT ist.
- 10 22. Satz mehrerer Anordnungen und eine mit jeder Anordnung des Satzes mehrerer Anordnungen verbundenen Server-Anordnung zum Schutz mehrerer Programme vor einem unbefugten Zugriff durch einen Prozeß,
wobei jede Anordnung einen Prozessor aufweist, der derart eingerichtet ist, daß folgende Schritte durchführbar sind:
- 15 - jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils ein Adreßraum zugeordnet,
- jedem zu schützenden Programm und/oder jeder zu schützenden Datei ist jeweils eine Prozeß-Datei zugeordnet,
- in einer Prozeß-Datei ist gespeichert, welcher Prozeß oder
- 20 welche Prozesse in dem jeweiligen Adreßraum ablaufen darf oder dürfen,
- während des Ablaufs eines zu schützenden Programms und/oder einer zu schützenden Datei wird für einen Prozeß, der auf den Adreßraum des zu schützenden Programms und/oder der zu schützenden Datei zugreifen will überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeß-Datei angegeben ist,
- 25 - für den Fall, daß der zugreifende Prozeß in der Prozeß-Datei angegeben ist, wird der zugreifende Prozeß gestartet oder weitergeführt, und
- 30 - sonst wird ein Alarmsignal generiert und an die Server-Anordnung gesendet,
und wobei die Server-Anordnung einen Prozessor aufweist, der derart eingerichtet ist, daß abhängig von mindestens einem empfangenen Alarmsignal eine vorgegebene Aktion ausgelöst
- 35 wird.

Zusammenfassung

Verfahren, Anordnung sowie ein Satz mehrerer Anordnungen zum Schutz mehrerer Programme und/oder mehrerer Dateien vor einem unbefugten Zugriff durch einen Prozeß

Es wird jedem zu schützenden Programm jeweils ein Bereich und eine Prozeß-Datei zugeordnet. In einer Prozeß-Datei ist gespeichert, welcher Prozeß oder welche Prozesse in dem jeweiligen Bereich ablaufen darf oder dürfen. Während des Ablaufs des Programms wird für einen Prozeß, der auf das Programm zugreifen will überprüft, ob der zugreifende Prozeß in der entsprechenden Prozeßdatei angegeben ist. Der zugreifende Prozeß wird nur ausgeführt, wenn er in der Prozeßdatei angegeben ist.

Signifikante Figur 1

FIG 1

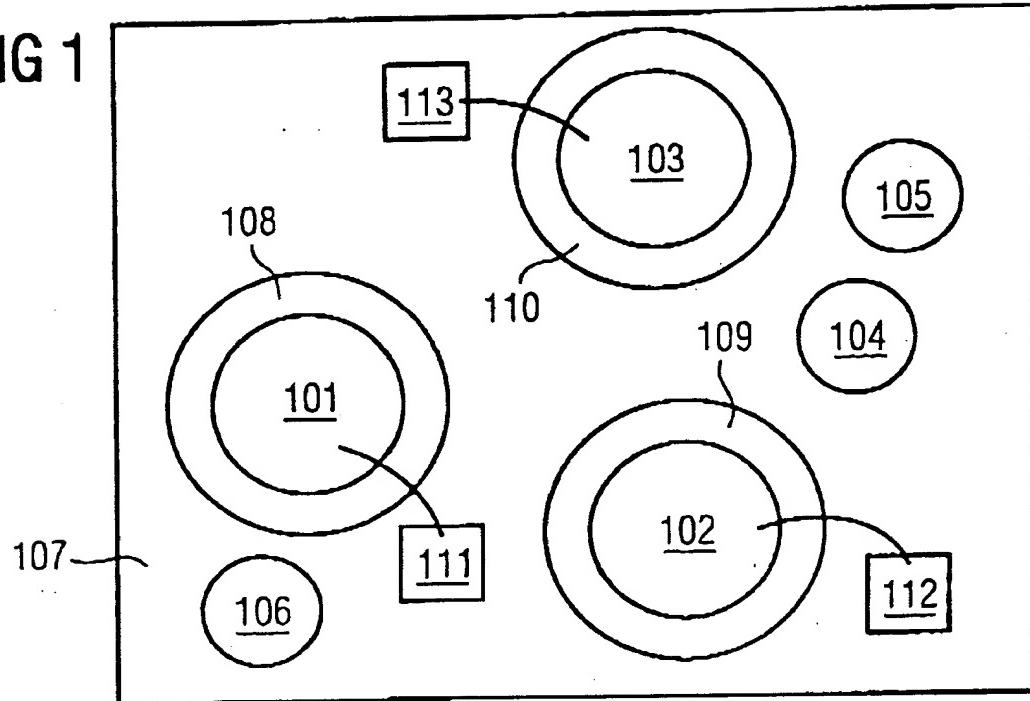
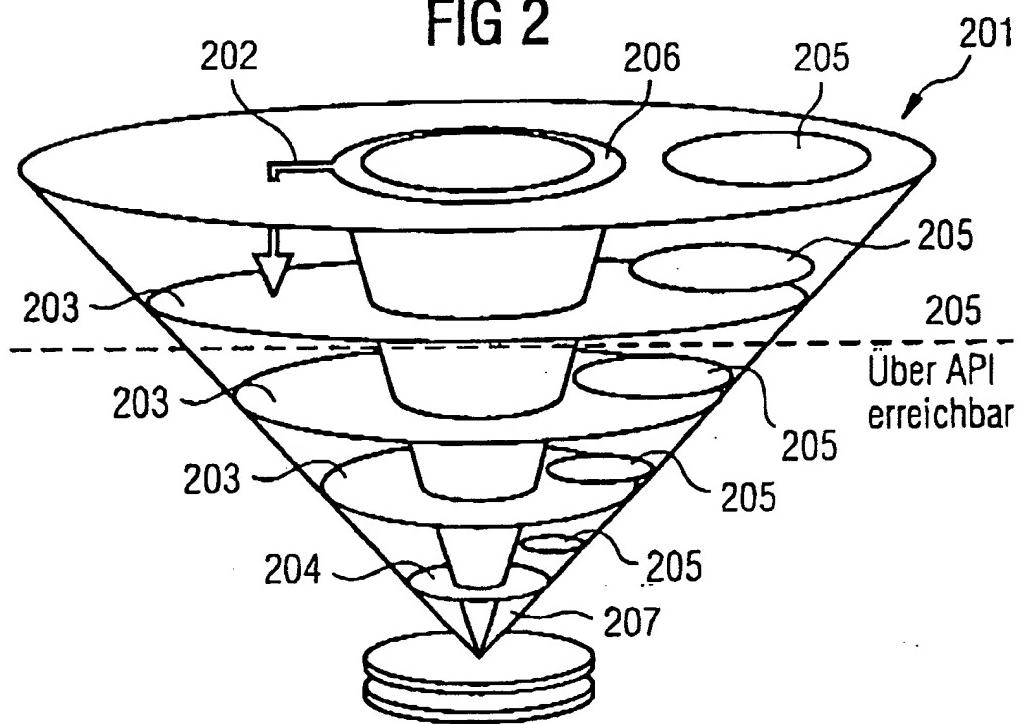


FIG 2



09/763029

2/6

FIG 3

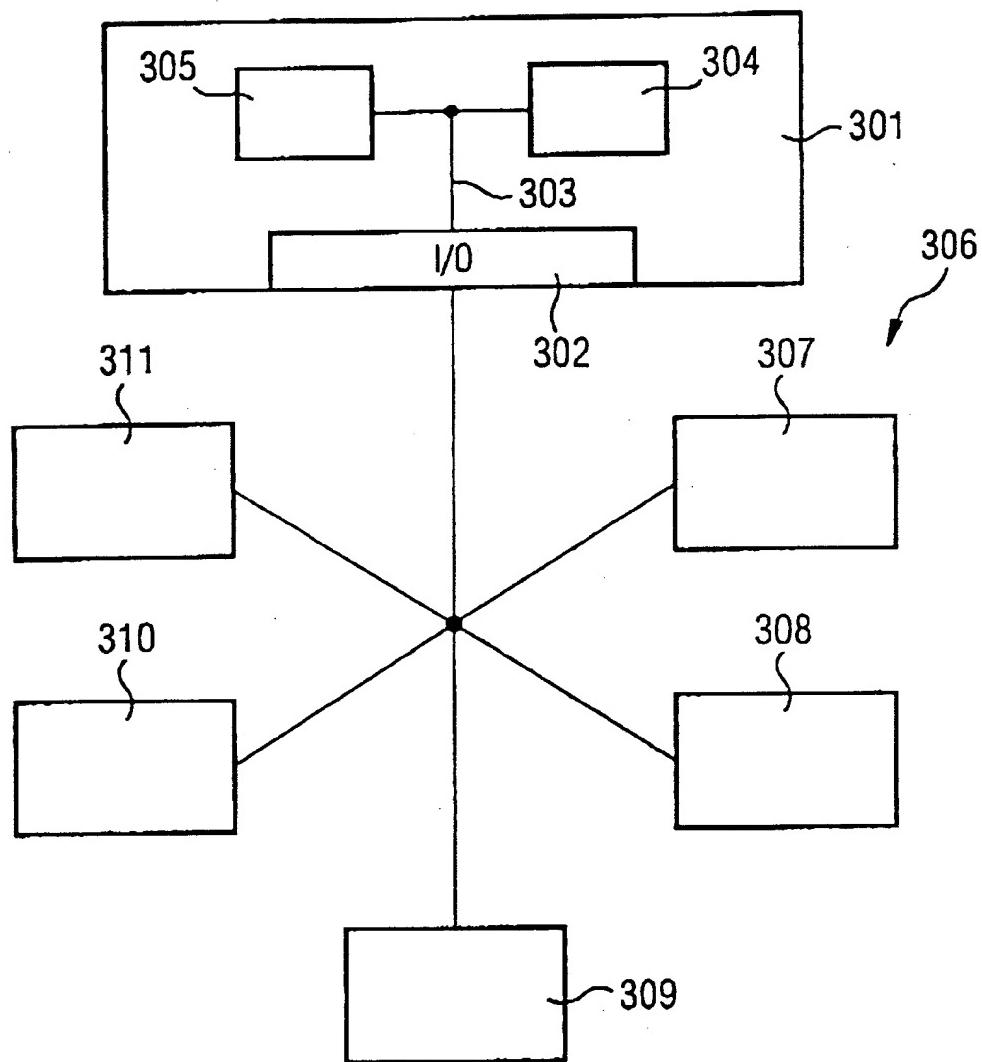


FIG 4

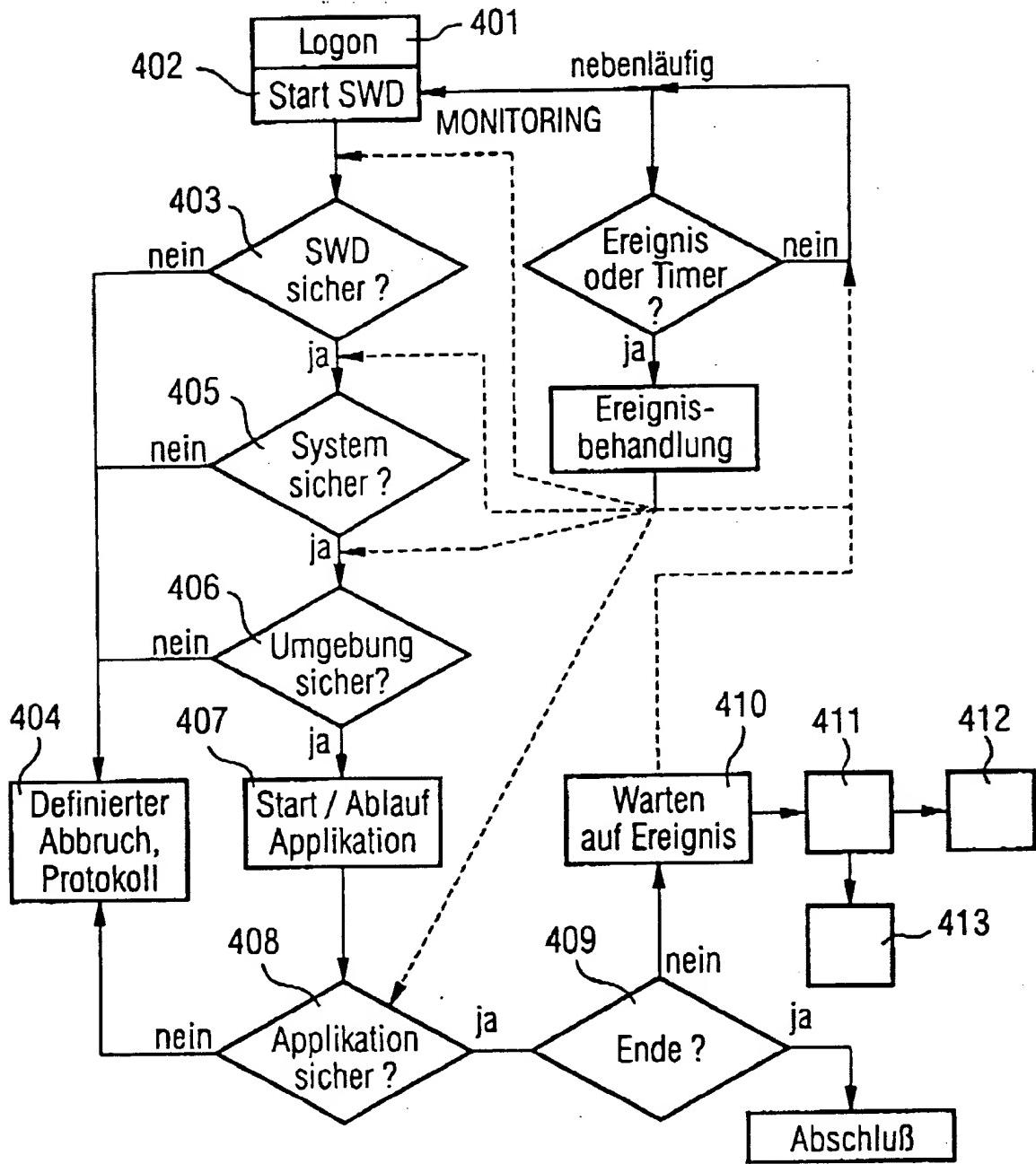


FIG 5

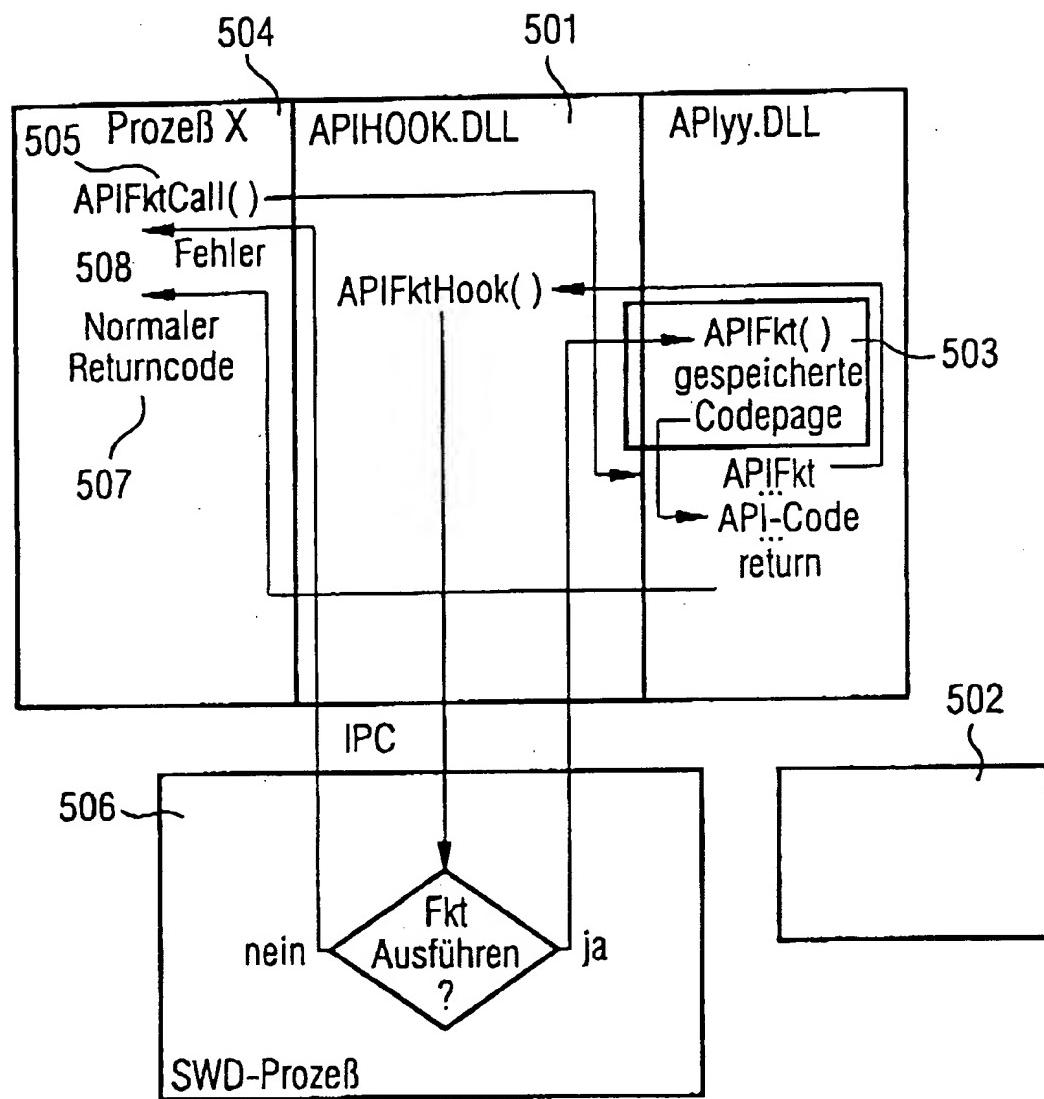
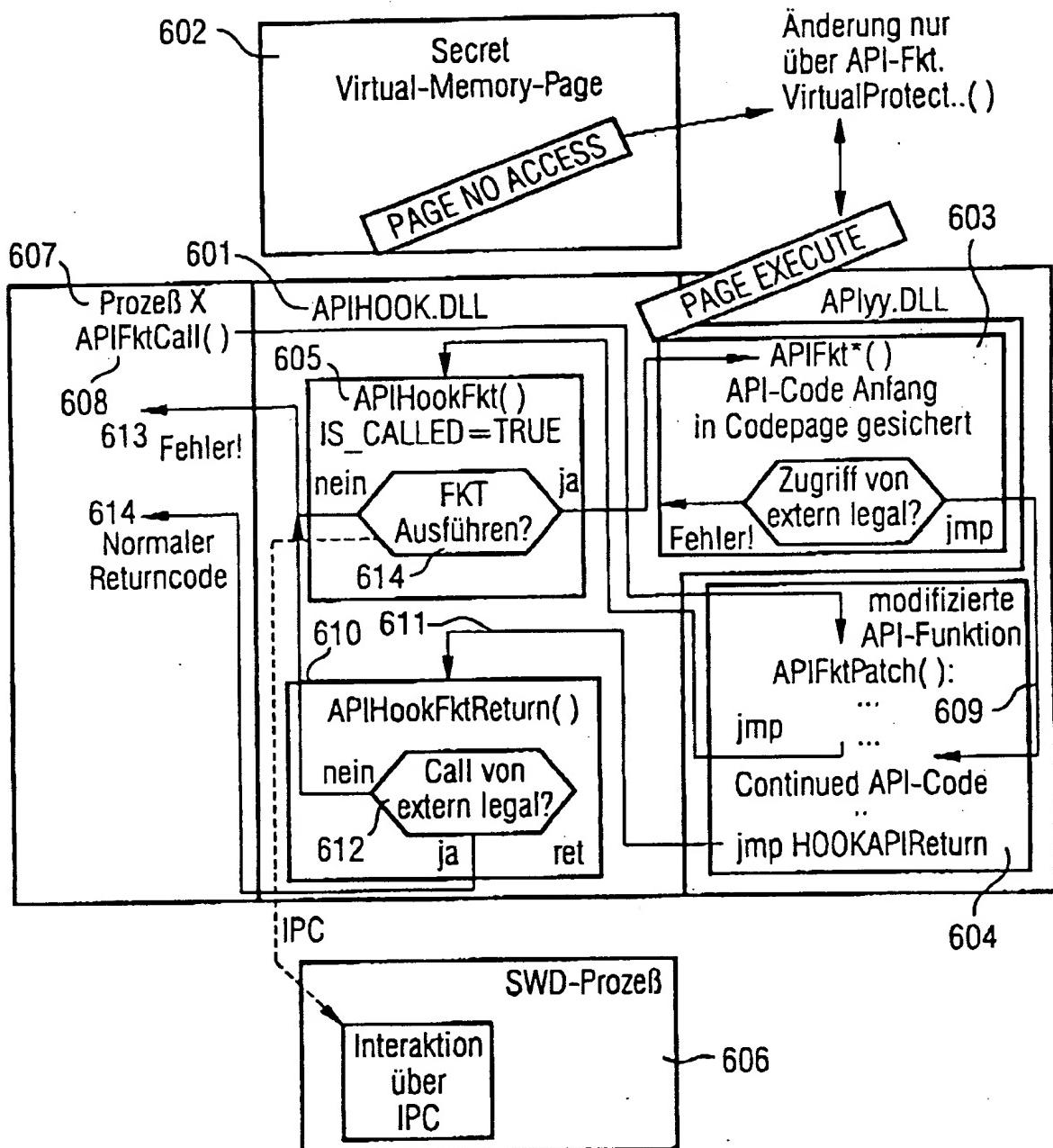
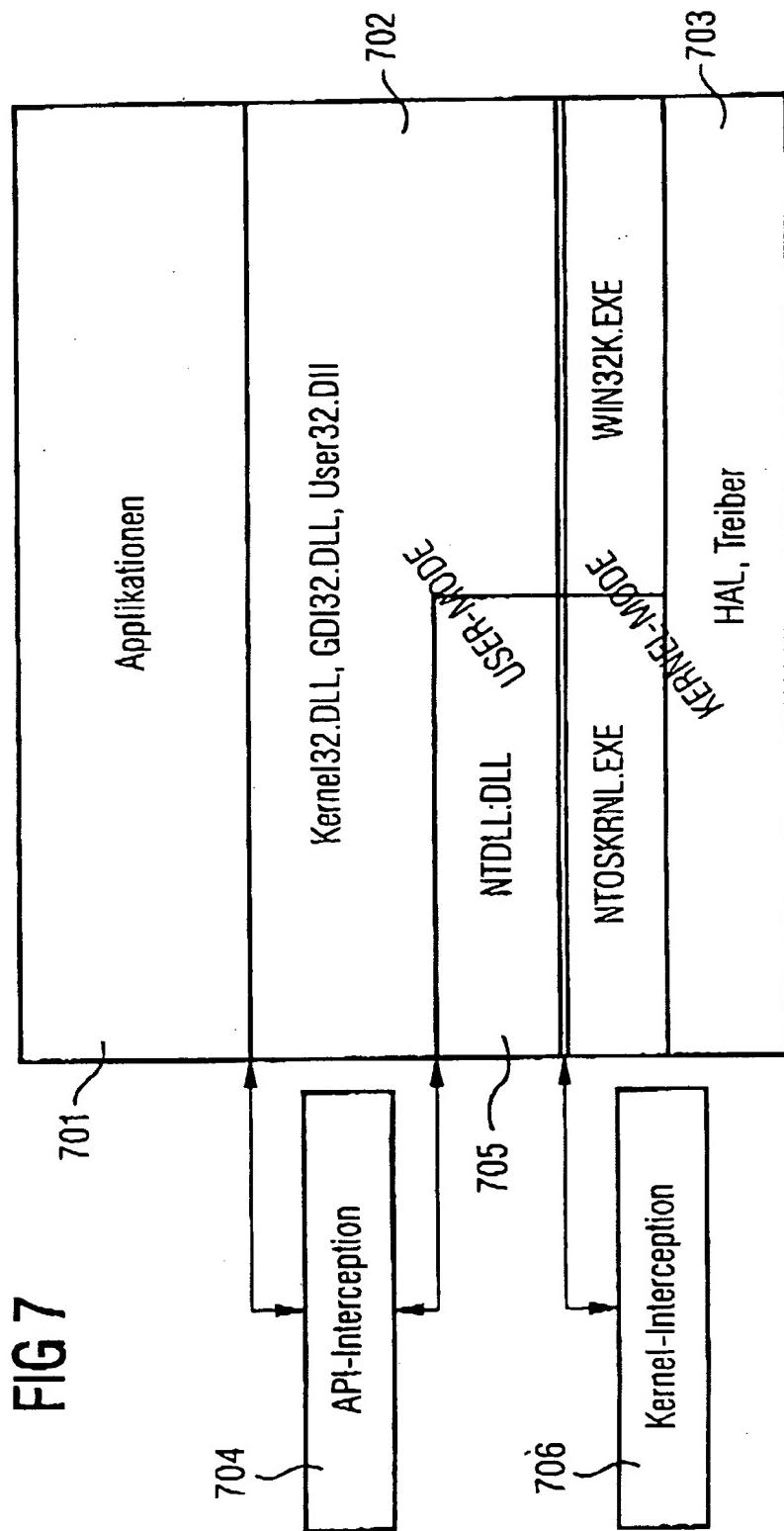


FIG 6





**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- BLACK BORDERS**
- IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- FADED TEXT OR DRAWING**
- BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- SKEWED/SLANTED IMAGES**
- COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- GRAY SCALE DOCUMENTS**
- LINES OR MARKS ON ORIGINAL DOCUMENT**
- REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.